🛠️ Ejercicio Práctico – Nivel Básico

📌 **Título:** *Implementación Básica de Autenticación y Autorización en una Aplicación Web*

🎯 Objetivo del ejercicio:

Implementar un sistema sencillo de **autenticación** con **contraseñas** y **autorización básica** en función de roles (como Administrador y Usuario común). Los estudiantes aprenderán a crear un sistema donde los usuarios puedan autenticarse, y luego se les asigne un acceso diferente según su rol.

📝 Escenario:

Imagina que estás desarrollando una aplicación web para gestionar contenido en un blog. El sistema tiene dos tipos de usuarios:

- **Administrador**: Puede ver, editar y eliminar cualquier publicación.
- **Usuario común**: Solo puede ver las publicaciones.

Tu tarea es implementar un sistema básico que permita a los usuarios registrarse, iniciar sesión con una contraseña y que luego el acceso a las publicaciones esté restringido según su rol.

🔍 Tu tarea:

Paso 1 – Implementar la autenticación básica con contraseñas:

1. **Crear una página de registro** donde los usuarios puedan ingresar su nombre de usuario y contraseña.
2. **Almacenar la contraseña de forma segura** utilizando un **algoritmo de hashing** (puedes usar **bcrypt** o un algoritmo similar para evitar almacenar contraseñas en texto claro).

3. **Crear una página de inicio de sesión** donde los usuarios puedan ingresar sus credenciales (nombre de usuario y contraseña). Al validar sus credenciales, deben ser autenticados y redirigidos a su página principal.

Paso 2 – Crear roles de usuario:

1. Al registrar un usuario, asigna un **rol** por defecto: **Usuario común**.

2. Crea una opción para que el **Administrador** asigne roles a los usuarios en la base de datos (puede ser al momento de la creación del usuario o mediante un panel de administración).

Paso 3 – Restringir el acceso según el rol:

1. **Administrador**: Los usuarios con rol de **Administrador** pueden acceder a todas las publicaciones del blog y tienen la opción de crear, editar y eliminar publicaciones.

2. **Usuario común**: Los usuarios con rol de **Usuario común** solo pueden ver las publicaciones pero no pueden modificarlas ni eliminarlas.

Paso 4 – Pruebas de seguridad:

1. Asegúrate de que un **Usuario común** no pueda acceder a las funciones que son exclusivas para los **Administradores** (como la edición o eliminación de publicaciones).

2. Asegúrate de que solo los **Usuarios autenticados** puedan ver el contenido del blog (es decir, implementa una verificación básica de autenticación para acceder a las publicaciones).

Paso 5 – Verificación y pruebas:

1. **Prueba el flujo de registro y inicio de sesión**: Asegúrate de que los usuarios puedan registrarse, iniciar sesión correctamente y acceder a sus respectivas funcionalidades.

2. **Prueba de acceso según roles**: Asegúrate de que el acceso a las funcionalidades esté correctamente restringido según el rol del usuario (solo los **Administradores** deben poder crear, editar y eliminar publicaciones).

✅ Resultado esperado:

- Un sistema funcional de **registro e inicio de sesión** con contraseñas **seguras** (utilizando hashing).
- Un sistema de **roles de usuario** básico (Administrador y Usuario común).
- **Restricción de acceso** a las funcionalidades según el rol del usuario.

📄 Entrega sugerida:

1. **Código fuente** de la implementación de autenticación y autorización.
2. **Capturas de pantalla** del registro de usuario, inicio de sesión y restricciones de acceso.
3. **Informe detallado** de cómo se implementaron la autenticación, los roles de usuario y las restricciones de acceso.

🧰 Herramientas recomendadas:

- **Node.js** con **Express.js** (o cualquier otro framework backend que prefieras).
- **bcrypt** o **argon2** para el hashing de contraseñas.
- **MongoDB** o **MySQL** para la gestión de usuarios y roles.
- **Postman** o **Insomnia** para probar las rutas de la API.

📝 Sugerencia para extender el ejercicio (opcional):

- **Autenticación multifactor (MFA)**: Implementa un sistema de autenticación más seguro que requiera un segundo factor (como un código enviado por correo electrónico o SMS).
- **Protección contra ataques comunes**: Implementa medidas de seguridad contra **CSRF** o **XSS** para mejorar la robustez del sistema.
